

# Some Hello World examples for ESPHome

By [jean-claude.feltes@education.lu](mailto:jean-claude.feltes@education.lu) 3.3.2023

## 1. Common part of the code for the examples

I have used a common part of code for all the examples, something like this:

```
substitutions:
  devicename: <name>

esphome:
  name: $devicename

esp8266:
  board: d1_mini or pico, see text

logger:
  level: DEBUG

api:
  password: !secret api_password

ota:
  password: !secret ota_password

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

ap:
  ssid: "Fallback Hotspot"
  password: !secret ap_password

manual_ip:
  static_ip: 192.168.0.33
  gateway: 192.168.0.100
  subnet: 255.255.255.0
```

(With SSID and password in the file secrets.yaml)

Some of the examples are for a Raspi Pico. Here you find instructions to use ESPHome with Pico:  
<https://www.youtube.com/watch?v=oQpEWwXqnbM>

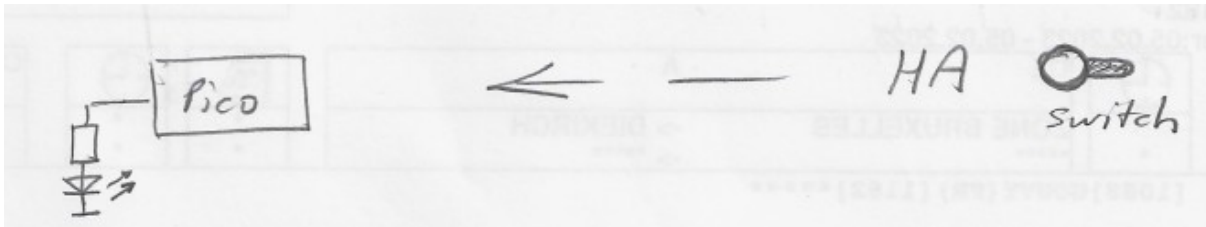
However this worked for me only over the Web interface on the host machine, which makes it very slow. That's why I switched to a D1 mini for the other examples so I could use the command line interface as described here:

[http://staff.ltam.lu/feljc/electronics/homeassistant/ESPHome\\_commandline.pdf](http://staff.ltam.lu/feljc/electronics/homeassistant/ESPHome_commandline.pdf)

## 2. Purely configurational examples

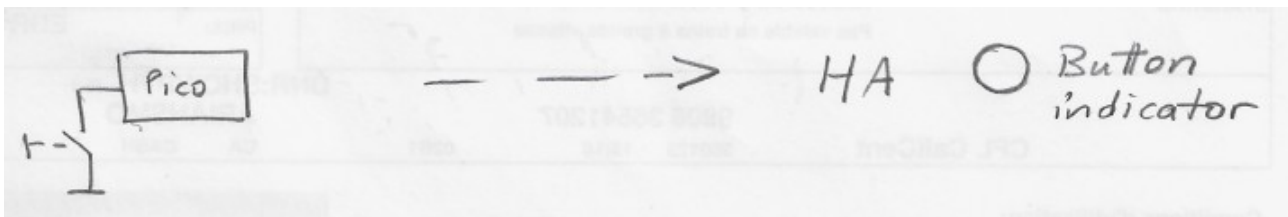
The first examples use only configuration of sensors or actors without any real programming.

### 2.1. HA switches Pico LED on and off



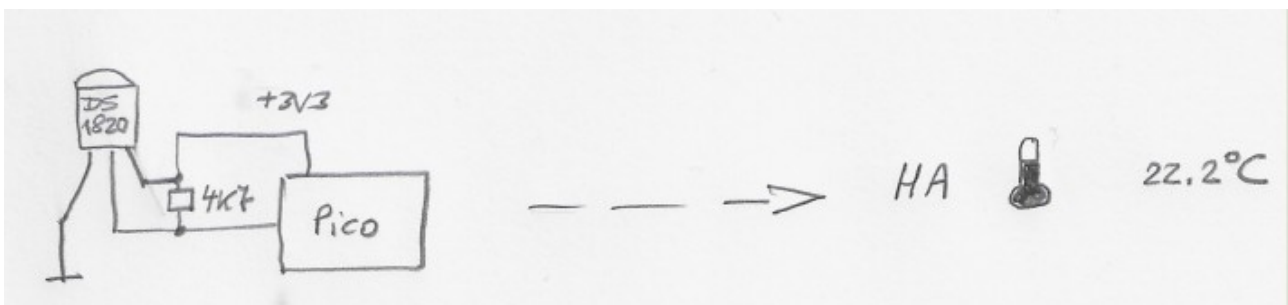
```
switch:
- platform: gpio
  name: "Pico LED"
  pin:
    number: GPIO0
```

### 2.2. HA indicates state of a hardware button (or digital sensor)



```
binary_sensor:
- platform: gpio
  name: "Button"
  pin:
    number: GPIO16
    inverted: True
    mode: INPUT_PULLUP
```

### 2.3. HA indicates temperature



```
dallas:
  - pin: GPIO17
    update_interval: 1s

sensor:
  - platform: dallas
    index: 0
    name: "Pico DS18B20"
```

### 3. Easy Programming examples

The following examples use some kind of programming (or at least a beginning of this) so that the controller reacts to events.

This behavior is available even when the connection to HA is lost. It can be compared to programming the controller in C++ (Arduino) or Micropython.

Advantages:

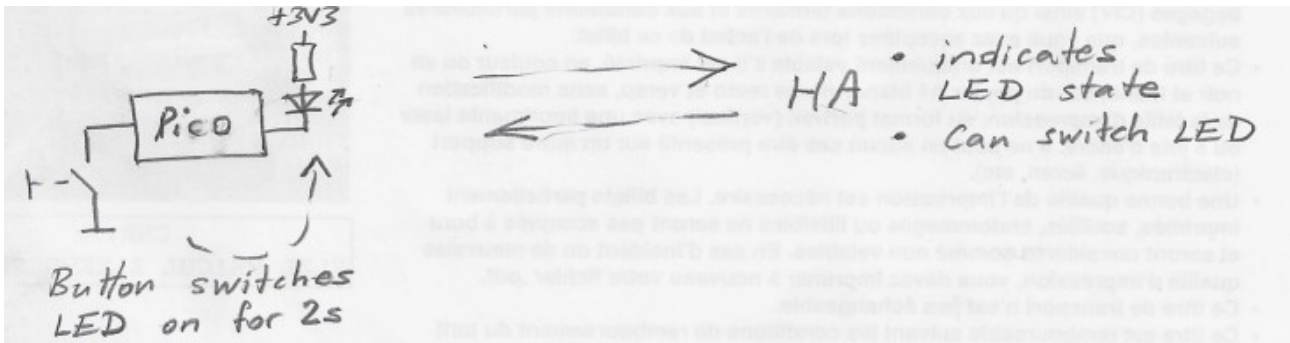
The WiFi link to HA is included without any programming, and it seems to be very reliable.

Disadvantage:

You have to learn a new “programming language” in YAML, which can do a lot of things, but doesn’t seem to offer as much control as programming the controller yourself.

<https://esphome.io/guides/automations.html#>

#### 3.1. Program Pico so that button switches LED on for 2s



```
switch:
  - platform: gpio
    name: "Pico LED"
    pin:
      number: GPIO00
      inverted: true
    id: led1

binary_sensor:
  - platform: gpio
    name: "Button"
    pin:
      number: GPIO16
      inverted: True
      mode: INPUT_PULLUP
    on_press:
```

```

then:
  - switch.toggle: led1
  - delay: 2s
  - switch.toggle: led1

```

Instead of `switch.toggle`, `switch.turn_on` and `switch.turn_off` can also be used.

### 3.2. Program a LED to light up in a certain temperature range

```

switch:
  - platform: gpio
    name: "Pico LED"
    pin:
      number: GPIO0
      inverted: true
    id: led1

dallas:
  - pin: GPIO17
    update_interval: 1s

sensor:
  - platform: dallas
    index: 0
    name: "Pico DS18B20"
    on_value_range:
      - above: 25.0
        then:
          - switch.turn_on: led1
      - below: 23.0
        then:
          - switch.turn_off: led1

```

### 3.3. Automations and Lambdas

<https://esphome.io/guides/automations.html#templates-lambdas>

To do some real programming you have to use lambda functions.

In lambdas you're effectively writing C++ code.

! lambda tells ESPHome that the following block is supposed to be interpreted as a lambda, or C code.

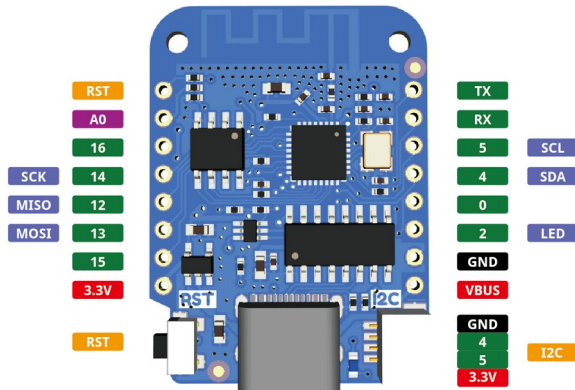
A “| - “ character combination tells the YAML parser to treat the following **indented (!)** block as plain text. Without it, the YAML parser would attempt to read the following block as if it were made up of YAML keys

With `if (...)` { ... } `else` { ... } we create a *condition*.

Finally, `id(...)` is a helper function that makes ESPHome fetch an object with the supplied ID

---

The following examples are for D1mini, but they should also work with a Pico.

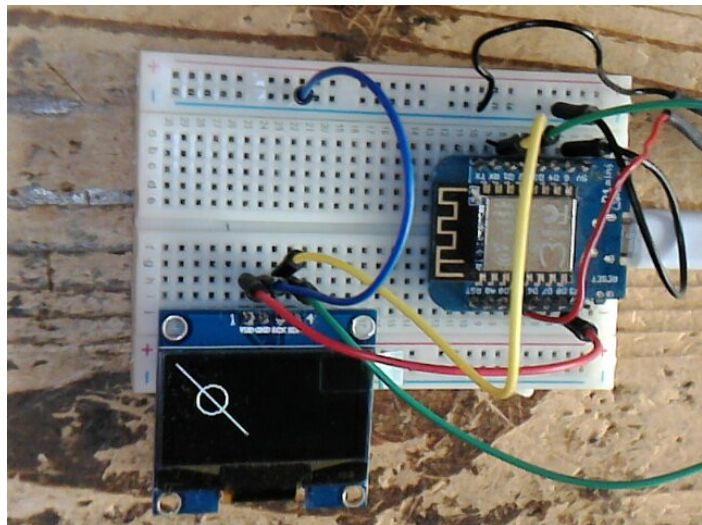


### 3.4. OLED-Displays showing static images and text

A simple example drawing a static image consisting of a line and a circle:

```
# Display:
i2c:
  sda: GPIO4
  scl: GPIO5

display:
- platform: ssd1306_i2c
  model: "SH1106 128x64"
  address: 0x3C
  lambda: |-
    it.fill(COLOR_OFF);
    it.line(0, 0, 50, 50);
    it.circle(25, 25, 10);
```



The drawing worked fine, but I had problems connecting to the WiFi and to HA after adding this part of the code. After a lot of test and research on the Internet I came to the conclusion that the cause must be a timing problem.

The code is not executed from top to bottom of the YAML file. First comes the OLED part, then the WiFi connection.

I finally found that the example worked fine when I added an update time statement to the display code:

```
display:
- platform: ssd1306_i2c
  model: "SH1106 128x64"
  address: 0x3C
  update_interval: 10s
```

```
lambda: |-
  it.fill(COLOR_OFF);
  it.line(0, 0, 50, 50);
  it.circle(25, 25, 10);
  it.filled_circle(60, 25, 10);
  it.draw_pixel_at(60, 25, COLOR_OFF);
  it.rectangle(100, 50, 20, 12, COLOR_ON);
```

I could reduce the interval to 5s without problem, at 2s there was a reconnect and finally it worked. Probably it is a good idea to choose an interval that is not too short.

There are other more advanced techniques to solve this problem, see part 2 of this document that hopefully will be ready soon.

**Writing text** is also possible, even using True Type Fonts.

#### Static text:

```
font:
- file:
  type: gfonts
  family: Roboto
  weight: 500
  id: font2
  size: 16

display:
- platform: ssd1306_i2c
  model: "SH1106 128x64"
  address: 0x3C
  update_interval: 5s
  lambda: |-
    it.fill(COLOR_OFF);
    it.print(0, 0, id(font2), "Hello World!");
```

In this example a Google font is used that is automatically downloaded.

We can also use any TTF font that we download to a subfolder font of our working directory.

### 3.5. OLED displays showing dynamic text

#### Writing sensor values:

```
sensor:
- platform: adc
  pin: VCC
  name: $devicename VCC Voltage
  id: vcc

time:
- platform: homeassistant
  id: esptime

i2c:
  sda: GPIO4
  scl: GPIO5
```



```

scan: false

font:
- file: "font/courier.ttf"
  id: myfont
  size: 16

display:
- platform: ssd1306_i2c
  model: "SH1106 128x64"
  address: 0x3C
  update_interval: 5s
  lambda: |-
    it.print(0, 0, id(myfont), "Hello World!");
    it.printf(0, 20, id(myfont), "VCC=%2.3fV", id(vcc).state);
    it.strftime(0, 40, id(myfont), "%H:%M:%S", id(esptime).now());

```

This example uses a courier font that we have downloaded to the font folder before compilation.

The display shows the supply voltage gotten from the adc sensor, and the time transmitted from Home Assistant. The formatting is done with the % operator and with strftime, as it is done in C or Python.

Info:

[https://esphome.io/cookbook/display\\_time\\_temp\\_oled.html](https://esphome.io/cookbook/display_time_temp_oled.html)

## 4. Tips

- Don't use names with underscores like "hello\_1"  
<https://esphome.io/guides/faq.html#why-shouldn-t-i-use-underscores-in-my-device-name>

- Use debug log while developing:

```

logger:
  level: DEBUG

```

There should be a line in the USB log like

```
[14:47:20][D][api.connection:918]: Home Assistant 2023.1.7
(192.168.0.154): Connected successfully
```

when the device is found by HA.

- Use entity names consisting of devicename and entity name, like "mydevice temperature"  
Why? Entities belonging to one device are easily found in an alphabetical list in HA.

This is done best with substitutions:  
substitutions:

```
devicename: halli4

esphome:
  name: $devicename

...
# switch LED from HA:
switch:
  - platform: gpio
    name: $devicename LED
    pin:
      number: GPIO2
      inverted: True
```

- Use static IP addresses:

```
wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

ap:
  ssid: "Fallback Hotspot"
  password: !secret ap_password

manual_ip:
  static_ip: 192.168.0.30
  gateway: 192.168.0.100
  subnet: 255.255.255.0
```

- Maybe the device is not found by HomeAssistant?
  - Check DEBUG log (Is there a compilation error? Is it connecting to WiFi?...)
  - Try to add it manually (This is relatively easy for static IP addresses):  
see next paragraph

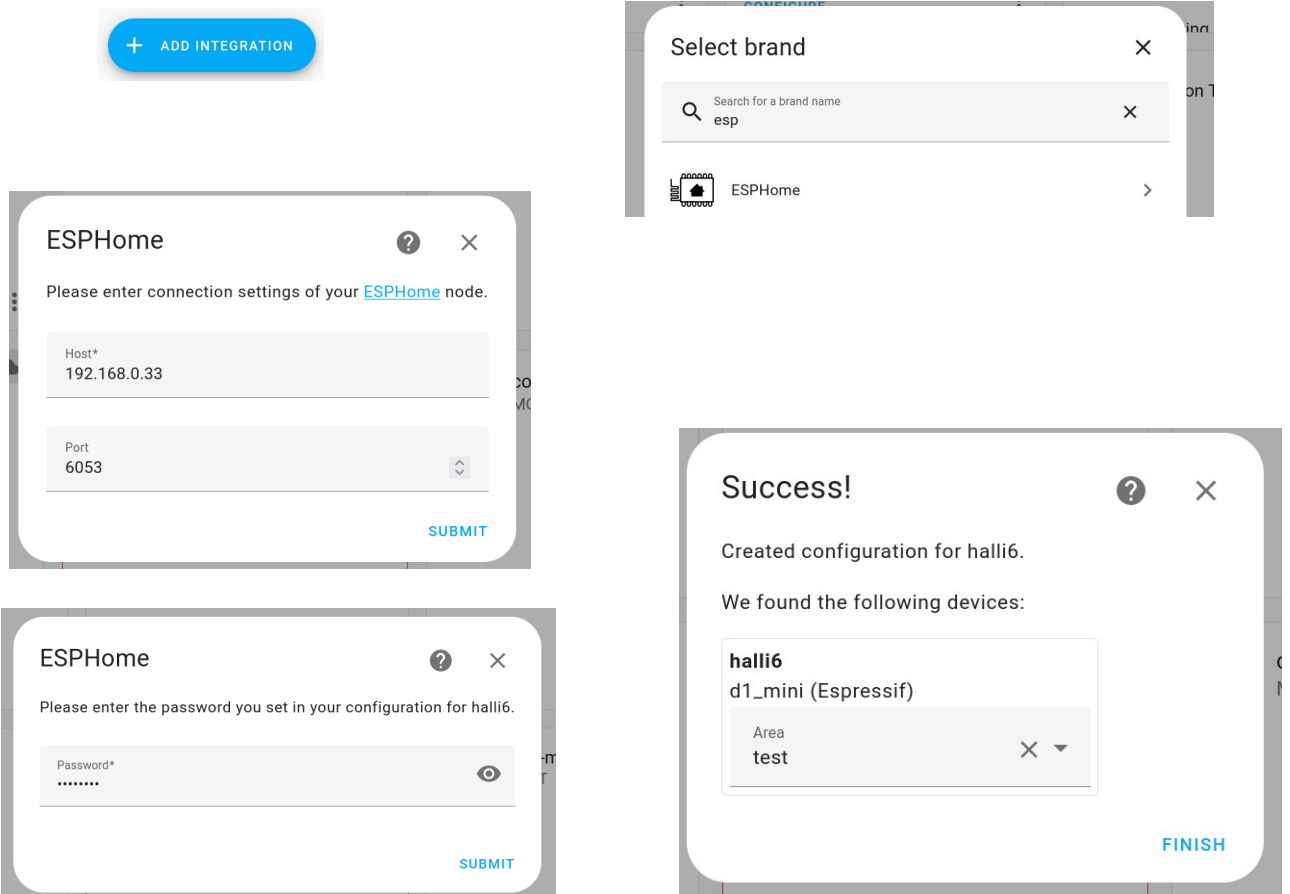


## 5. Manually add devices that are not found by HA

Maybe this problem arises when using the command line to compile. (?)

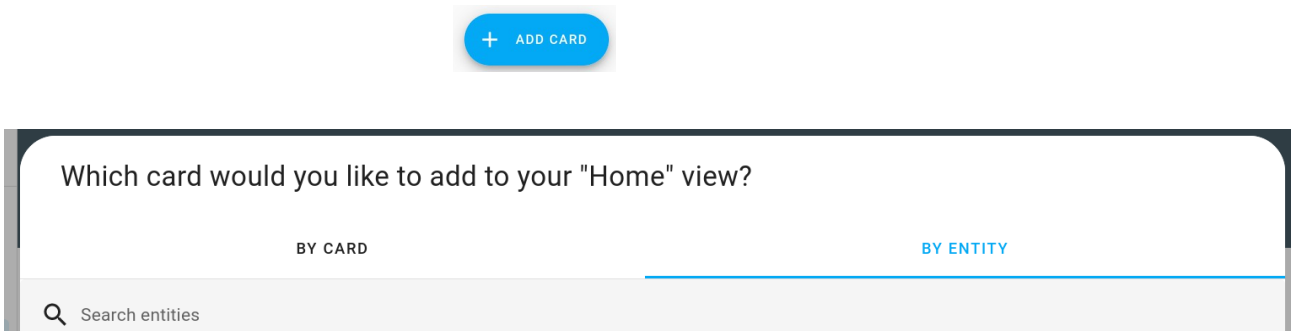
I succeeded in finding the devices manually:

- Menu Settings – Devices & Services






After this we still have to add the entities to view to a card in a dashboard:

In a dashboard:



The entities are listed alphabetically (if not realized earlier, here you see the importance of assigning a device name to the entities (see tips))

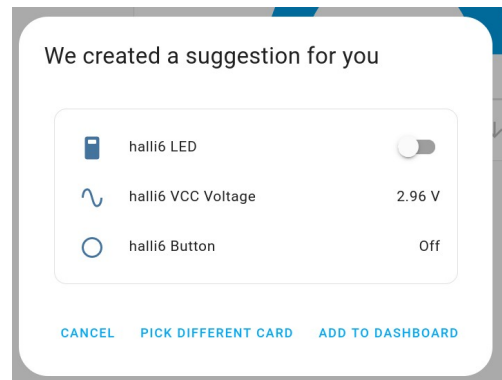
Check the entities to show in the card:

<input checked="" type="checkbox"/>		halli6 VCC Voltage sensor.halli5_vcc_voltage
<input checked="" type="checkbox"/>		halli6 Button binary_sensor.halli6_button
<input checked="" type="checkbox"/>		halli6 LED switch.halli6_led

Continue

Accept proposal of the system:

Add to dashboard – ready.



## 6. Sources

Guides:

<https://esphome.io/guides/>

Components index:

<https://esphome.io/index.html>

FAQ:

<https://esphome.io/guides/faq.html#tips-for-using-esphome>

Display:

<https://esphome.io/components/display/index.html#>

<https://esphome.io/index.html#display-components>

Other info:

<https://esphome.io/guides/configuration-types.html>

<https://www.youtube.com/watch?v=a3iay-g1AsI>

<https://tech.scargill.net/my-esphome-adventure/>